Structured Programming Language (CSE113)
Lecture #04
Arrays in C programming

An **array** is a collection of data that holds fixed number of values of same type. For example, if you want to store marks of 100 students, you can create an array for it.

float marks [100];

The **size** and **type** of arrays cannot be changed after its declaration.

Arrays are of two types:

1. One-dimensional arrays
2. Multidimensional arrays

How to declare an array in C?

data_type array_name[array_size];

This is called a one-dimensional array. The **array_size** must be an integer constant greater than zero and type can be any valid C data type.

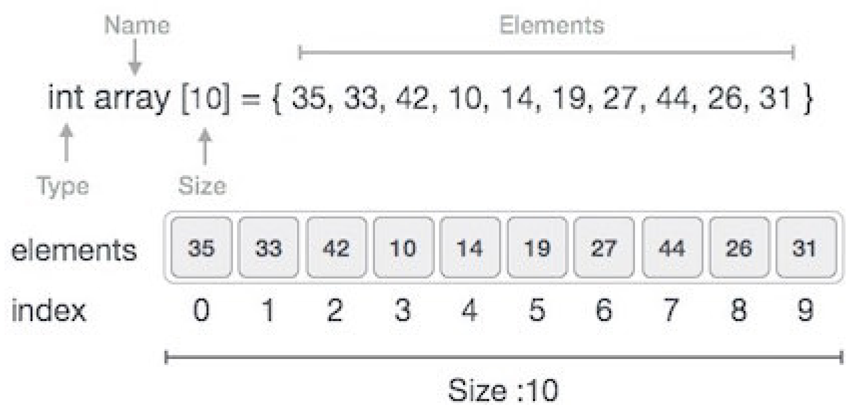
For example:

float marks[100];

Here, we declared an array, *marks*, of floating-point type and size 100. Meaning, it can hold 100 floating-point values.

Following are the important terms to understand the concept of Array.

- **Element** − each item stored in an array is called an element.
- **Index** − each location of an element in an array has a numerical index, which is used to identify the element.



As per the above illustration, following are the important points to be considered.

- Array length = 10, First Index = 0, Last Index = 9
- Index starts with 0.
- Array length is 10 which means it can store 10 elements.
- Each element can be accessed via its index. For example, we can fetch an element at index 6 as 27.

Why do we need arrays?

We can use normal variables (v1, v2, v3, ...) when we have small number of objects, but if we want to store large number of instances, it becomes difficult to manage them with normal variables. The idea of array is to represent many instances in one variable.

Basic Operations

Following are the basic operations supported by an array.

- **Traverse** − print all the array elements one by one.
- **Insertion** − Adds an element at the given index.
- **Deletion** − Deletes an element at the given index.
- **Search** − Searches an element using the given index or by the value.
- **Update** − Updates an element at the given index.

Q1. Print 1-dimension array elements with index.

Source Code:

```
#include<stdio.h>
int main()
{
   int a[5] = {7,2,3,9,5};
   int i;
   for(i = 0; i < 5; i++)
      printf("a[%d] = %d\n",i,a[i]);
   return 0;
}
```

Output:

```
a[0] = 7
a[1] = 2
a[2] = 3
a[3] = 9
a[4] = 5
```

Source Code:

```
#include<stdio.h>
#define MAX 10

int main()
{
   int a[MAX] = {7,2,3,9,5};
   int i;
   for(i = 0; i < MAX; i++)
      printf("a[%d] = %d\n",i,a[i]);
   return 0;
}
```

Output:

```
a[0] = 7
a[1] = 2
a[2] = 3
a[3] = 9
a[4] = 5
a[5] = 0
a[6] = 0
a[7] = 0
a[8] = 0
a[9] = 0
```

- ✓ **A one-dimensional array is like a list.**

Q2. Write a program to find average marks obtained by a class of 5 students in a test.

Source Code:

```
#include<stdio.h>
main()
{
   int avg, i, sum=0;
   int marks[5] ;    //array declaration
   for ( i = 0 ; i < 5 ; i++ )
   {
      printf ( "Enter marks: " ) ;
      scanf ( "%d", &marks[i] ) ;    //store data in array
   }
   for ( i = 0 ; i < 5 ; i++ )
      sum = sum + marks[i] ;    //read data from an array
   avg = sum / 5 ;
   printf ( "\nAverage marks = %d\n", avg ) ;
}
```

Output:

```
G:\C_Bkp\Desktop\C\average
Enter marks: 10
Enter marks: 20
Enter marks: 30
Enter marks: 40
Enter marks: 50

Average marks = 30
```

Q3. Write a program to find largest element of an array.

Source Code:

```
#include<stdio.h>
int main()
{
   int a[100];
   int i,n;
   printf("Enter total number of elements(1 to 100): ");
   scanf("%d",&n);
   // Stores number entered by the user
   for(i = 0; i < n; i++){
      printf("Enter number %d: ",i+1);
      scanf("%d",&a[i]);
   }
    // Loop to store largest number to arr[0]
   for(i = 0; i < n; i++){
      if(a[0] < a[i]){
         a[0] = a[i];
      }
   }
   printf("Largest Element is: %d\n",a[0]);
}
```

Output:

G:\C_Bkp\Desktop\C\arrayLargestElement.exe

```
Enter total number of elements(1 to 100): 6
Enter number 1: 10
Enter number 2: 300
Enter number 3: 1
Enter number 4: 500
Enter number 5: 66
Enter number 6: 2
Largest Element is: 500
```

Q4. Print 2-dimension array elements.

Source Code:

```
#include<stdio.h>

int main()
{
   int a[3][2] = {  {10,20}, {30,40}, {50,60}  };
   int i,j;

   for(i = 0; i < 3; i++){
      for(j = 0; j < 2; j++){
         printf("%d\t",a[i][j]);
         }
      printf("\n");
   }
}
```

Output:

```
10      20
30      40
50      60
```

Q5. Print 2-dimension array elements with index.

Source Code:

```
#include<stdio.h>

int main()
{
   int a[3][2] = { {10,20}, {30,40}, {50,60}  };
   int i,j;

   for(i = 0; i < 3; i++){
      for(j = 0; j < 2; j++){
         printf("a[%d][%d] = %d\t",i,j,a[i][j]);
         }
      printf("\n");
   }
}
```

Output:

```
a[0][0]=10      a[0][1]=20
a[1][0]=30      a[1][1]=40
a[2][0]=50      a[2][1]=60
```

- ✓ **A two-dimensional array is like a table.**

Q6. 2-dimension array summation

Source Code:

```
#include<stdio.h>
int main()
{
   int a[3][2] = {  {5,0}, {1,2}, {2,3}  };
   int b[3][2] = {  {1,2}, {2,3}, {3,6}  };
   int c[3][2];
   int i, j;

   for(i = 0; i < 3; i++){     //no. of rows
      for(j = 0; j < 2; j++){   //no. of columns
         c[i][j] = a[i][j] + b[i][j];
         printf( "%d\t", c[i][j] );
      }
      printf("\n");
   }
   return 0;
}
```

Output:

```
"C:\Users\User\Desktop\c programming practice\array\array.exe"
6       2
3       5
5       9

Process returned 0 (0x0)   execution time : 0.013 s
Press any key to continue.
```